10.48047/jocaaa.2026.35.03.02

Infrastructure as Code (IaC) Drift Detection: State Management Using Event-Driven Architecture

Arun Harikrishnan
UNYBRANDS LLC, USA

Abstract

Infrastructure as Code has become fundamental for modern cloud infrastructure management, enabling declarative resource provisioning through version-controlled configuration files. Traditional drift detection mechanisms rely on polling-based workflows that create significant temporal vulnerabilities where unauthorized modifications remain undetected for extended periods. These detection delays expose organizations to security risks, compliance violations, and operational instability across multi-cloud environments. This article introduces an innovative event-driven architecture for continuous Infrastructure as Code drift detection that eliminates polling-based limitations through real-time monitoring capabilities. The proposed framework implements a sophisticated dual-layer detection system combining continuous cloud audit log analysis with extended Berkeley Packet Filter-based system monitoring to provide comprehensive visibility into infrastructure modifications across both control plane and data plane components. The dual-layer architecture correlates events between cloud provider APIs and operating system-level configurations to achieve enhanced detection accuracy while reducing false positive rates through cross-validation mechanisms. Experimental evaluation across multi-cloud testbed environments demonstrates substantial improvements in Mean Time to Detection while maintaining minimal system overhead and seamless integration with existing DevOps workflows. The event-driven system successfully identifies unauthorized infrastructure changes, shadow IT activities, and compliance violations within seconds of occurrence rather than hours or days typical of traditional polling mechanisms. Real-world deployment scenarios validate the framework's effectiveness across diverse organizational contexts, including regulatory compliance monitoring, emergency configuration change detection, and automated remediation workflows. The article demonstrates how event-driven drift detection transforms Infrastructure as Code tools from periodic deployment mechanisms into continuous enforcement frameworks for infrastructure immutability, security governance, and operational compliance.

Keywords: Event-Driven Architecture, Infrastructure As Code, Drift Detection, Multi-Cloud Security, Real-Time Monitoring

1. Introduction

Context and Problem Statement

Infrastructure as Code has transformed cloud computing environments. Organizations now treat computing resources as software artifacts. This shift enables declarative infrastructure definition through version-controlled configuration files. Teams can test, validate, and deploy infrastructure through automated pipelines. IaC provides infrastructure consistency and reproducibility across distributed environments. It reduces operational overhead compared to manual resource provisioning methods.

Enterprise environments widely adopt IaC tools today. Terraform, AWS CloudFormation, and Azure Resource Manager are among the well-liked platforms. OpenTofu developed as a substitute in response to

10.48047/jocaaa.2026.35.03.02

recent licensing modifications in the HashiCorp environment. These tools enable complex infrastructure topology definitions. Organizations can specify virtual machines, networking components, storage systems, and security policies. Human-readable configuration languages abstract underlying cloud provider APIs. Teams maintain precise control over resource specifications and dependencies.

Current drift detection mechanisms rely on polling-based architectures. These systems execute periodic state comparison operations regularly. Teams typically schedule Terraform plan commands or equivalent validation procedures. This approach creates systematic temporal vulnerabilities in infrastructure governance. Drift windows allow unauthorized modifications to persist undetected. Configuration deviations and security policy violations can remain hidden during these periods. Polling intervals vary from minutes to hours based on organizational preferences. Longer intervals reduce computational overhead but increase exposure risks [1].

Security implications of prolonged drift windows extend beyond configuration inconsistencies. These vulnerabilities encompass compliance framework violations and unauthorized access vector creation. Resource provisioning outside governance boundaries poses significant risks. Shadow IT activities represent particularly concerning scenarios. Development teams sometimes circumvent approval processes to address operational requirements. Emergency configuration changes during incident response bypass standard change control processes. These modifications restore service availability but create drift conditions. Often undetectable for years are illegal modifications to infrastructure.

Motivation and Research Gaps

Contemporary research focuses on improving polling-based detection accuracy rather than addressing temporal limitations. Existing polling mechanisms achieve high detection rates when drift conditions are eventually discovered. However, systematic delays between drift occurrence and detection remain problematic. These delays represent critical vulnerabilities in current IaC tools and research approaches.

Traditional IaC state management exhibits several blind spots in infrastructure governance. Cloud control plane monitoring captures API-level resource modifications effectively. These systems lack visibility into operating system-level configuration changes. System-level monitoring tools provide detailed individual resource configuration data. However, they lack integration with declarative IaC state definitions. These gaps create opportunities for sophisticated attackers or configuration errors to remain undetected [2].

Regulatory compliance requirements intensify the need for real-time infrastructure governance capabilities. Organizations have to quickly spot configuration changes affecting compliance position. Government, financial services, and healthcare industries are under rising legal examination. Conventional survey methods offer insufficient responsiveness for compliance systems. Organizations require comprehensive audit trails and rapid response mechanisms for security-relevant modifications.

Research Contribution

This research introduces event-driven architecture for continuous Infrastructure as Code drift detection. The approach eliminates temporal vulnerabilities inherent in traditional polling methods. Real-time monitoring of infrastructure change events occurs across multiple observation layers. The methodology implements sophisticated dual-layer detection frameworks. Continuous cloud audit log analysis is combined with extended Berkeley Packet Filter-based system monitoring. This provides comprehensive, immediate visibility into infrastructure modifications. Both the cloud control plane and data plane components receive monitoring coverage.

The dual-layer detection methodology advances beyond existing single-source monitoring approaches. Changes across cloud provider APIs correlate with operating system-level configurations. This correlation achieves enhanced detection accuracy while reducing false positive rates. Cross-validation of change

10.48047/jocaaa.2026.35.03.02

events improves reliability significantly. The integrated approach addresses critical blind spots in contemporary IaC tools. Seamless compatibility with existing deployment workflows and organizational change management processes remains intact.

Event-driven drift detection transforms Infrastructure as Code tools fundamentally. Traditional periodic deployment mechanisms become continuous enforcement frameworks. Infrastructure immutability, security governance, and operational compliance receive proactive management. Organizations establish responsive infrastructure management capabilities. Unauthorized changes trigger responses within seconds rather than hours or days. Security posture and operational reliability improve substantially across enterprise cloud environments.

Paper Organization

Section 2 provides a comprehensive literature review content. The evolution of Infrastructure as Code methodologies receives detailed examination. Current drift detection approaches undergo thorough analysis. Event-driven architecture principles applied to infrastructure management challenges receive exploration. Section 3 presents detailed methodology and system architecture specifications. The proposed dual-layer event-driven detection framework includes implementation details. Cloud control plane monitoring and eBPF-based data plane observation receive technical coverage. Section 4 describes experimental evaluation approaches with quantitative results. Event-driven approach effectiveness spans multiple deployment scenarios and organizational use cases. Section 5 concludes with research contributions discussion and practical implications for enterprise infrastructure management. Future research directions in continuous infrastructure governance require identification.

2. Literature Review and Background

Code as Infrastructure Development

Originating from difficulties in managing large-scale distributed computing systems, infrastructure as Code arose. Conventional infrastructure management depended rather on manual procedures. Administrators performed server configurations individually across multiple machines. Shell scripts and basic automation tools handled routine tasks. This manual approach became problematic as cloud adoption increased rapidly. Organizations require repeatable provisioning methods with proper version control.

The movement from imperative to declarative infrastructure management transformed operational approaches completely. Imperative systems demanded explicit instructions for every infrastructure change. Administrators wrote detailed procedures for each system modification. Declarative systems let teams define desired end states without implementation details. Teams describe target infrastructure configurations rather than construction steps. This transformation aligned with software development practices, treating infrastructure like application code. Infrastructure configurations adopted version control, automated testing, and continuous integration practices.

Contemporary IaC platforms implement varied approaches to state management and resource provisioning. Terraform maintains centralized state repositories tracking resource metadata and interdependencies. CloudFormation uses stack-based deployments with comprehensive template versioning. Ansible combines configuration management concepts with infrastructure provisioning through organized playbooks. Each platform handles drift detection and state synchronization using different strategies. These strategies vary in implementation complexity, system scalability, and workflow integration capabilities.

10.48047/jocaaa.2026.35.03.02

Current industry approaches to drift detection rely on scheduled validation processes. Organizations execute periodic Terraform plan operations to identify discrepancies between intended and actual configurations. Validation schedules range from hourly to daily execution based on risk tolerance and operational requirements. Remediation approaches include manual operations team intervention or automated correction through infrastructure pipeline re-execution. Effectiveness varies significantly based on organizational automation maturity.

Event-Driven Architecture in Infrastructure Management

Event-driven architecture establishes theoretical foundations for building reactive systems in distributed cloud platforms. EDA enables loose component coupling through asynchronous messaging mechanisms. This design philosophy enables highly scalable, robust systems that fit shifting operating environments. Over lengthy periods, event sourcing keeps thorough audit logs of system state changes. These patterns are used by companies to recreate past conditions and evaluate operational change patterns.

Modern EDA solutions in DevOps settings place deployment automation above continuous monitoring skills. Automatic build and deployment processes are started by continuous integration systems using webhook triggers. Automatically responding to source code modifications, infrastructure provisioning uses coordinated testing and deployment procedures. These implementations demonstrate EDA benefits for coordinated system operations while lacking comprehensive state monitoring features. Current implementations miss real-time infrastructure change detection and automated response capabilities [3].

Integration approaches between cloud audit systems and automation workflows require additional exploration in existing literature. Cloud platforms provide comprehensive audit logging through dedicated services. These audit streams primarily serve compliance documentation and forensic investigation rather than operational automation. Existing integration focuses on log collection and historical analysis instead of immediate automated responses. Organizations encounter significant challenges in implementing real-time integration between audit systems and operational processes.

10.48047/jocaaa.2026.35.03.02

Tool Category	State Management Approach	Drift Detection Method
Declarative IaC	Centralized state tracking	Periodic plan execution
Cloud-Native Tools	Stack-based deployment	Template comparison
Configuration Management	Playbook-based state	Agent-based monitoring

Table 1: IaC Tools State Management Comparison. [3]

Drift Detection Methodologies

Infrastructure drift encompasses multiple categories of configuration deviations impacting system security and regulatory compliance. Configuration drift represents changes to resource parameters that deviate from original infrastructure specifications. Security drift includes unauthorized modifications to access permissions, network policies, or encryption configurations. Compliance drift covers changes violating regulatory standards or organizational governance policies. Each drift type requires distinct detection approaches and targeted remediation strategies.

Existing detection systems demonstrate significant temporal constraints that compromise infrastructure governance effectiveness. Polling-based approaches introduce delays between drift occurrence and system identification. These delays fluctuate based on polling intervals and overall system complexity. Batch validation processes cannot deliver immediate responses to critical security breaches or compliance violations. Organizations face prolonged exposure windows where unauthorized modifications remain invisible.

Immutable infrastructure concepts treat infrastructure elements as replaceable rather than modifiable components. This philosophy demands complete component replacement instead of in-place modifications when changes become necessary. Enforcement relies on automated deployment processes that block manual alterations and maintain deployment consistency. Existing tooling offers limited enforcement for infrastructure immutability across diverse multi-cloud deployments.

Drift Category	Detection Scope	Temporal Limitation
Configuration Drift	Resource parameter changes	Hours to days delay
Security Drift	Access control modifications	Extended exposure window
Compliance Drift	Policy violation detection	Batch processing delays

Table 2: Drift Detection Methodology Classification. [3]

Cloud Observability and Monitoring

Cloud audit logging systems deliver comprehensive visibility into infrastructure operations across primary cloud platforms. AWS CloudTrail records detailed API interactions, including user authentication, precise timestamps, and resource modification specifics. Azure Activity Log documents subscription-level events and comprehensive resource change tracking. GCP Cloud Audit Logs preserve administrative action records and sensitive data access histories. These platforms generate substantial event volumes suitable for real-time processing implementations.

eBPF-based observability represents innovative kernel-level monitoring for containerized and virtualized environments. eBPF programs execute within kernel space without requiring kernel alterations or system downtime. This technology provides granular monitoring of system calls, network traffic, and filesystem operations. Implementation areas include security monitoring, performance measurement, and compliance

10.48047/jocaaa.2026.35.03.02

validation across diverse computing platforms. eBPF solutions offer unprecedented system behavior visibility [4].

Integration complexities between the control plane and data plane monitoring generate significant observability gaps in current infrastructure approaches. Control plane monitoring targets cloud provider APIs and high-level resource management operations. Data plane monitoring tracks application activities and detailed system operations within individual resources. Effective event correlation across these monitoring layers demands sophisticated processing and correlation capabilities. Organizations struggle with unified monitoring implementations spanning multiple observability domains.

Research Gaps

Contemporary IaC drift detection demonstrates insufficient real-time capabilities that undermine security and operational effectiveness. Most commercial and open-source solutions depend on periodic validation instead of continuous monitoring approaches. This constraint generates temporal vulnerabilities, allowing unauthorized changes to persist undetected. Real-time detection demands fundamental architectural modifications to existing IaC platforms and infrastructure management practices.

Operating system configuration changes receive inadequate attention in traditional IaC drift detection implementations. Cloud control plane monitoring effectively captures resource provisioning and configuration through API tracking. These approaches completely miss system-level changes occurring within individual compute instances. Essential configuration files, service specifications, and security policies can change without generating cloud provider audit events.

Comprehensive frameworks integrating multiple detection layers remain unavailable in current literature and commercial offerings. Available tools target individual monitoring domains instead of integrated multi-layer approaches. Multi-layer detection demands advanced correlation algorithms and event processing beyond current tool capabilities. Missing standardized integration patterns restrict comprehensive drift detection effectiveness across complex enterprise environments.

3. Methodology and System Architecture

Event-Driven Drift Detection Framework

The dual-layer detection system creates comprehensive architectural foundations combining real-time cloud control plane monitoring with detailed data plane observation. This design removes time delays found in traditional infrastructure drift detection methods. The system uses distributed event processing patterns for instant infrastructure deviation identification. Existing organizational workflows and deployment pipelines maintain compatibility during implementation. The framework functions through continuous event stream analysis instead of periodic polling.

Component integration connects cloud audit streams with eBPF monitoring using advanced correlation engines analyzing events across multiple observation layers. The integration maintains synchronized timing between control plane API operations and matching system-level changes within compute resources. Event correlation algorithms find relationships between cloud provider operations and underlying system modifications across the infrastructure. This allows complete detection of coordinated infrastructure changes spanning multiple monitoring domains.

Event processing pipeline design emphasizes real-time state comparison analyzing incoming events against current infrastructure state definitions. Terraform state files and similar configuration management systems store these definitions across different environments. The pipeline uses parallel processing architectures, handling multiple concurrent events while preserving ordering requirements for dependent operations. Stream processing algorithms classify events by resource types, modification extent, and

10.48047/jocaaa.2026.35.03.02

security implications during analysis. The design includes configurable filtering, reducing noise from routine activities while maintaining complete coverage of security-relevant infrastructure changes. Real-time event processing architectures create essential foundations for effective cloud infrastructure monitoring with optimized design patterns and improved performance capabilities [5].

Cloud Control Plane Detection Layer

Cloud audit log streaming implementation creates persistent connections to major cloud provider audit systems across multiple platforms. These systems include AWS CloudTrail, Azure Activity Log, and Google Cloud Audit Logs for complete coverage. The streaming architecture uses high-throughput event ingestion to process multiple audit events per second effectively. End-to-end latency stays minimal for critical security events needing immediate attention. Filtering mechanisms analyze incoming audit streams, identifying infrastructure-relevant operations while removing administrative activities not affecting declared resource states.

API operation classification algorithms analyze audit events, determining their potential impact on infrastructure state consistency and security posture. Classification engines use machine learning models distinguishing between authorized infrastructure changes through established deployment pipelines. These models identify unauthorized modifications representing security threats or policy violations requiring immediate investigation. State deviation analysis algorithms compare detected changes against current IaC state definitions throughout comparison processes. This identification happens through resource identifier mapping, configuration parameter analysis, and dependency relationship validation across complex infrastructure topologies.

IaC state file integration needs sophisticated parsing and synchronization mechanisms maintaining current infrastructure state definitions across multiple tools and environments. The system uses real-time state file monitoring detecting updates to Terraform, OpenTofu, and CloudFormation definitions automatically. This ensures drift detection operates against current declared infrastructure specifications. Resource dependency mapping creates complete graphs of infrastructure relationships, enabling accurate impact assessment when changes occur. Event correlation techniques identify patterns of related changes, indicating coordinated infrastructure modifications or potential security incidents requiring immediate response.

Data Plane Detection Layer

eBPF program design emphasizes efficient system-level configuration monitoring, providing complete visibility into operating system changes within compute resources. This monitoring happens without significant performance overhead on running systems. Custom eBPF programs deploy dynamically to target instances through orchestration systems, ensuring consistent monitoring coverage across the infrastructure. The programs use selective monitoring approaches focusing on configuration-critical system components during execution. These approaches avoid routine system activities not significantly impact infrastructure security or compliance.

Kernel event filtering mechanisms capture file system modifications, network configuration changes, and process execution patterns indicating unauthorized system alterations outside established change control processes. File system monitoring tracks changes to critical configuration directories, service definition files, and security-related system components, potentially compromising infrastructure immutability principles. Network monitoring observes modifications to routing tables, firewall rules, and interface configurations creating security vulnerabilities or violating network security policies. Process monitoring identifies unauthorized service installations, configuration management tool executions, and privilege escalation attempts indicating security incidents. eBPF-based security monitoring delivers complete

10.48047/jocaaa.2026.35.03.02

system visibility for containerized environments with effective implementation strategies and detailed performance analysis capabilities [6].

Container runtime integration enables immutable infrastructure enforcement across containerized applications and orchestrated computing environments. The integration uses policy engines preventing unauthorized modifications to container configurations, image specifications, and runtime parameters deviating from declared infrastructure specifications. Operating system baseline comparison algorithms maintain reference configurations for each monitored resource throughout operational lifecycles. This enables precise detection of deviations from expected system states across diverse computing environments. Deviation detection algorithms use statistical analysis techniques to distinguish between normal system variations and significant configuration changes requiring investigation and remediation procedures.

Real-Time Processing Architecture

Event streaming infrastructure uses Apache Kafka clusters or cloud-native messaging services providing scalable, fault-tolerant event processing capabilities. These capabilities handle high-volume event streams from multiple monitoring sources simultaneously across distributed environments. The streaming architecture uses topic-based event organization, enabling parallel processing of different event types while maintaining ordering guarantees for related events. Message partitioning strategies ensure load balancing across processing nodes while preventing event loss during system failures or maintenance operations. The infrastructure supports configurable retention policies, maintaining event history for forensic analysis and compliance reporting requirements.

Stream processing implementation enables parallel drift analysis across multiple infrastructure components while maintaining consistency in state comparison operations. Processing algorithms use windowing techniques, grouping related events for coordinated analysis across complex infrastructure topologies. This enables the detection of complex drift scenarios involving multiple infrastructure changes occurring simultaneously. State reconciliation mechanisms resolve conflicts between different monitoring sources and handle scenarios where events arrive out of order due to network delays or system processing variations. The processing system uses exactly-once delivery semantics, ensuring accurate drift detection under challenging operational conditions.

Through standardized API interfaces, webhook methods, and message queue integrations, integration with current IaC processes enables easy inclusion in CI/CD pipelines and deployment automation systems. The integration architecture supports flexible configuration options accommodating diverse organizational requirements for automated versus manual response procedures. Integration adapters provide compatibility with popular DevOps tools and change management systems throughout implementation. This ensures drift detection capabilities enhance rather than disrupt existing operational workflows. Scalable distributed stream processing systems provide essential foundations for effective real-time infrastructure state management with complete scalability and reliability considerations [7].

Remediation and Response Framework

Automated remediation policies use configurable response procedures evaluating detected drift conditions against organizational risk tolerance, resource criticality assessments, and compliance framework requirements. Policy engines support complex rule definitions specifying different response procedures based on drift severity, affected resource types, and current operational contexts. Approval workflows integrate with existing organizational governance processes, ensuring automated responses align with established change management procedures and security policies. The framework supports immediate

10.48047/jocaaa.2026.35.03.02

automated responses for critical security incidents and human-approved procedures for complex infrastructure changes requiring careful analysis.

Standardized interfaces give all the detected drift conditions and recommended remediation actions, hence facilitating integration with current incident response and change management systems. Automated ticketing generation, stakeholder communication, and escalation procedures consistent with organizational incident response processes are supported by the integration. Change management integration ensures remediation activities receive appropriate documentation and approval oversight while maintaining rapid response capabilities for security-critical situations. The system maintains complete audit trails of all remediation activities, supporting compliance reporting and post-incident analysis procedures.

Rollback mechanisms implement automated infrastructure state restoration procedures, quickly reverting unauthorized changes to previously known good configurations. These mechanisms integrate with existing backup and versioning systems, ensuring reliable state restoration capabilities. Infrastructure state restoration procedures support granular resource-level rollbacks and complete environment-level restoration depending on incident scope and organizational requirements. Audit trail generation provides detailed documentation of all detected drift events, remediation actions, and system responses for complete compliance reporting and forensic analysis capabilities.

4. Implementation and Assessment

Experimental Arrangement

The evaluation framework assessed event-driven drift detection effectiveness across different cloud environments and infrastructure deployment scenarios. These scenarios matched typical enterprise operational patterns in modern organizations. The multi-cloud testbed included AWS, Azure, and Google Cloud Platform environments configured for extensive testing capabilities. Terraform and OpenTofu workflows underwent testing across different complexity levels and geographical distributions during evaluation. The evaluation infrastructure contained virtual machine instances, containerized application deployments, managed database services, networking components, and security policy configurations. Resources are spread across multiple geographical regions, simulating realistic global enterprise infrastructure deployment patterns.

IaC deployment scenarios represented authentic organizational use cases from simple applications to complex multi-service architectures. These architectures had intricate dependency relationships and cross-cloud resource integration requirements reflecting real-world complexity. Terraform configurations contained comprehensive resource definitions covering compute instances, storage systems, networking infrastructure, security groups, and managed services. These configurations match modern cloud-native applications deployed in enterprise environments. OpenTofu deployments matched Terraform scenarios, ensuring comparative evaluation validity while testing compatibility with the emerging open-source alternative.

Controlled drift introduction methods evaluated detection capabilities across comprehensive change categories. These categories covered unauthorized resource provisioning, security configuration modifications, network topology alterations, service parameter adjustments, and compliance policy violations. Each drift scenario replicated authentic operational situations, including emergency response procedures, shadow IT resource creation, and automated scaling operations. Configuration management tool operations modifying system state outside established IaC workflows were included. The baseline polling-based detection system used traditional scheduled validation approaches with standard Terraform

10.48047/jocaaa.2026.35.03.02

plan and Terraform show operations executed at regular intervals. Multi-cloud environments create significant performance evaluation challenges for infrastructure monitoring approaches with different deployment scenarios needing comprehensive analysis [8].

Detection Accuracy and Performance Metrics

Mean Time to Detection comparisons showed substantial improvements in infrastructure change identification capabilities, comparing event-driven approaches against traditional polling-based methods. The event-driven system showed consistent performance advantages across different change types, including resource provisioning modifications, security policy alterations, network configuration changes, and service parameter adjustments. Detection latency measurements revealed significant reductions in time between actual infrastructure changes and system identification of drift conditions. Improvements were especially notable for security-critical modifications needing immediate attention and response from operations teams.

False positive and false negative rate analysis showed robust accuracy characteristics across different infrastructure change types and operational contexts during evaluation. The dual-layer detection approach effectively reduced false positives through cross-validation of events between cloud control plane monitoring and data plane observation systems. False negative rates stayed consistently low across all tested drift categories during evaluation. The system successfully identified subtle configuration changes that traditional polling approaches frequently missed due to timing constraints or insufficient monitoring coverage.

System resource utilization analysis revealed efficient operation characteristics for continuous monitoring components deployed across the testbed infrastructure. eBPF monitoring programs showed minimal impact on host system performance while providing comprehensive visibility into system-level configuration changes. Cloud audit log processing components maintained efficient resource consumption patterns under high-volume event conditions. Memory utilization patterns scaled predictably with infrastructure complexity while maintaining consistent performance characteristics across different deployment scenarios. Scalability evaluation across varying infrastructure complexity levels validated the system's ability to maintain detection accuracy and performance characteristics as infrastructure footprints expanded from simple deployments to complex multi-cloud environments. Event-driven monitoring systems need careful scalability assessment, ensuring optimal performance across different cloud infrastructure management scenarios [9].

Detection Approach	Response Time Characteristic	Resource Overhead Pattern
Traditional Polling	Scheduled intervals	Periodic resource spikes
Event-Driven Control Plane	Real-time response	Continuous low utilization
Dual-Layer Integration	Immediate correlation	Optimized resource usage

Table 3: Event-Driven vs. Polling-Based Detection Comparison. [9]

Real-World Scenario Testing

Shadow IT detection capabilities achieved exceptional accuracy in identifying unauthorized infrastructure provisioning and modification activities across different organizational contexts and user behavior patterns. The system successfully correlated user authentication information, approval workflow data, and resource provisioning events identifying policy violations within minimal time periods. Integration with existing identity management systems enabled automated response workflows immediately restricting unauthorized activities while generating appropriate administrative notifications. Comprehensive audit

10.48047/jocaaa.2026.35.03.02

documentation supported compliance and forensic analysis purposes across multiple organizational requirements.

Emergency configuration change identification demonstrated the system's effectiveness in rapidly detecting and documenting critical infrastructure modifications performed under operational pressure during incident response scenarios. The event-driven methodology offered rapid visibility into emergency adjustments while still keeping exhaustive audit tracks. These audit trails satisfy post-incident analysis and compliance reporting needs under several legal systems. Automated remediation systems allowed for quick reaction to emergency changes, leading either to security breaches or compliance infractions. The system preserved operational changes necessary for incident resolution while maintaining security oversight.

Compliance drift detection evaluation focused on regulatory frameworks, including SOC 2, PCI DSS, and HIPAA requirements, demanding rigorous infrastructure governance and audit trail maintenance. The system successfully identified configuration changes impacting compliance posture across all tested regulatory frameworks. Automated documentation generation and notification capabilities supported continuous compliance monitoring requirements. Real-time violation detection significantly reduced compliance risk exposure periods compared to traditional periodic audit approaches. Integration testing with existing DevOps toolchains validated seamless incorporation of event-driven monitoring capabilities into established organizational workflows and approval processes. Webhook integrations, API connectivity, and message queue implementations provided flexible integration options accommodating different organizational tool preferences and operational procedures.

Performance Analysis

Event processing latency measurements revealed consistent response times for critical infrastructure change detection across all tested scenarios and operational conditions. Optimization strategies included event filtering algorithms prioritizing security-critical changes while maintaining comprehensive monitoring coverage for routine operational activities. Processing latency remained stable under varying load conditions while scaling appropriately with infrastructure complexity and event volume increases. These characteristics support growing organizational environments with expanding infrastructure footprints and operational requirements.

System overhead assessment showed minimal impact from eBPF monitoring and cloud audit log processing components on overall infrastructure performance and operational efficiency. eBPF programs maintained negligible CPU utilization while providing comprehensive system-level visibility across different computing environments. These environments included virtual machines, containers, and managed service platforms commonly used in enterprise deployments. Cloud audit log processing exhibited efficient resource consumption patterns, scaling appropriately with organizational size and infrastructure complexity. Most deployment scenarios did not require dedicated computing resources for optimal operation.

Comparative analysis of detection coverage revealed significant advantages for event-driven methods over traditional polling approaches across multiple evaluation criteria and operational scenarios. The dual-layer monitoring approach achieved comprehensive visibility into infrastructure changes that single-source monitoring systems frequently missed. These limitations often resulted from coverage gaps or timing constraints inherent in traditional approaches. Detection coverage improvements proved particularly significant for complex multi-layer infrastructure modifications and sophisticated scenarios attempting to evade traditional monitoring approaches. Cost-benefit analysis indicated favorable return on investment characteristics for organizations with moderate to large infrastructure footprints across

10.48047/jocaaa.2026.35.03.02

different industry sectors. Infrastructure monitoring expenses remained reasonable compared to traditional approaches while providing substantially enhanced detection capabilities. Technology implementation needs comprehensive cost-benefit analysis evaluating organizational value and return on investment across different deployment scenarios [10].

Case Study Results

Production environment deployment experiences across multiple organizational contexts provided valuable insights into real-world implementation challenges and effectiveness metrics for event-driven drift detection systems. Organizations reported substantial improvements in infrastructure visibility and control capabilities while maintaining compatibility with existing operational procedures and organizational governance frameworks. Implementation experiences highlighted the importance of comprehensive planning for integration with existing monitoring and alerting systems. This planning maximized operational benefits while minimizing disruption during deployment phases across different organizational environments.

Quantitative analysis of Mean Time to Remediation improvements showed significant reductions in duration between drift detection and successful remediation across different incident categories. Security risk reduction metrics showed measurable improvements in organizational security posture through enhanced detection capabilities and automated response mechanisms. These methods kept unapproved modifications from continuing in production settings over time. The enhancements resulted in lower compliance risk exposure and improved audit readiness over several regulatory systems and organizational demands.

Integration challenges primarily centered on coordination with existing change management processes and organizational approval workflows. These workflows required modification accommodating real-time detection and response capabilities without disrupting established operational procedures. Solutions included configurable policy engines adapting to different organizational requirements while maintaining essential security and compliance monitoring capabilities. Enterprise-scale implementations required careful consideration of event processing capacity and network bandwidth requirements. These considerations ensured optimal performance under peak operational conditions across different organizational environments.

User experience feedback from DevOps teams indicated high satisfaction with enhanced infrastructure visibility and automated response capabilities. These capabilities reduced manual monitoring overhead while improving overall operational efficiency across different organizational contexts. Workflow integration feedback emphasized the value of flexible integration options accommodating different toolchain preferences and established operational procedures. The integration occurred without requiring fundamental changes to existing development and deployment practices that organizations had already optimized for their specific requirements and operational constraints.

Implementation Aspect	Traditional Approach	Event-Driven Solution
Detection Coverage	Limited visibility	Comprehensive monitoring
Operational Integration	Manual processes	Automated workflows
Compliance Monitoring	Periodic audits	Continuous validation

Table 4: Real-World Implementation Results. [10]

10.48047/jocaaa.2026.35.03.02

Conclusion

The implementation of event-driven architecture for Infrastructure as Code drift detection represents a transformative advancement in cloud infrastructure governance that addresses fundamental limitations in traditional polling-based monitoring systems. The dual-layer detection framework successfully demonstrates comprehensive infrastructure change visibility through coordinated cloud control plane and data plane monitoring that eliminates temporal vulnerabilities inherent in periodic validation approaches. Experimental evaluation validates significant improvements in detection capabilities with immediate identification of unauthorized modifications, security policy violations, and compliance deviations that previously remained undetected for extended periods. The integration of continuous cloud audit log analysis with extended Berkeley Packet Filter-based system monitoring provides unprecedented visibility into infrastructure modifications across multiple observation layers while maintaining minimal performance impact on monitored systems. Real-world deployment experiences confirm the framework's practical effectiveness across diverse organizational contexts with seamless integration into existing DevOps workflows and change management processes. The event-driven system successfully transforms Infrastructure as Code tools from static deployment mechanisms into dynamic enforcement platforms that provide continuous governance for infrastructure immutability, security compliance, and operational integrity. Organizations implementing this framework achieve substantial reductions in Mean Time to Detection and Mean Time to Remediation while enhancing overall security posture through automated response capabilities and comprehensive audit trail generation. The dual-layer monitoring architecture addresses critical blind spots in existing infrastructure governance tools by correlating changes across cloud provider APIs and operating system configurations to provide complete visibility into infrastructure state modifications. Future enhancements may incorporate machine learning algorithms for intelligent anomaly detection and predictive drift identification capabilities that could further advance infrastructure security and compliance monitoring effectiveness. The event-driven drift detection framework establishes essential foundations for next-generation Infrastructure as Code tools that prioritize continuous enforcement over periodic validation, enabling organizations to maintain infrastructure consistency while supporting the dynamic requirements of modern cloud-native applications and distributed system architectures.

References

[1] Yevhenii Martseniuk et al., "Shadow IT risk analysis in public cloud infrastructure," CSDP-2024: Cyber Security and Data Protection, 2024. [Online]. Available: https://ceur-ws.org/Vol-3800/paper3.pdf

[2] Ritesh Kumar, "Event-Driven Architectures for Real-Time Data Processing: A Deep Dive into System Design and Optimization," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/391633680_Event-Driven_Architectures_for_Real-Time_Data_Processing_A_Deep_Dive_into_System_Design_and_Optimization

[3] Sandeep Rangineni, Arvind Kumar Bhardwaj, "ANALYSIS OF DEVOPS INFRASTRUCTURE METHODOLOGY AND FUNCTIONALITY OF BUILD PIPELINES," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/373037714_ANALYSIS_OF_DEVOPS_INFRASTRUCTURE_METHODOLOGY_AND_FUNCTIONALITY_OF_BUILD_PIPELINES

[4] Jin Her et al., "An In-Depth Analysis of eBPF-Based System Security Tools in Cloud-Native Environments," ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/395225966_An_In-Depth_Analysis_of_eBPF-Based_System_Security_Tools_in_Cloud-Native_Environments

10.48047/jocaaa.2026.35.03.02

[5] Siddharth Kumar Choudhary, "IMPLEMENTING EVENT-DRIVEN ARCHITECTURE FOR REAL-TIME DATA INTEGRATION IN CLOUD ENVIRONMENTS," IJCET, 2025. [Online]. Available: https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_16_ISSUE_1/IJCET_16_01_113.pdf

[6] Perumal Murugan, "Enhancing Container Security Using eBPF-Based Detection Mechanisms for Real-Time Threat Monitoring and System-Level Visibility," International Journal of Science, Engineering and Technology, 2024. [Online]. Available: https://www.ijset.in/wp-content/uploads/IJSET_V10_issue4_330.pdf

[7] Mitch Cherniack et al., "Scalable Distributed Stream Processing," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/220988218_Scalable_Distributed_Stream_Processing

[8] Jose M. Alcaraz Calero, Juan Gutiérrez Aguado, "Comparative analysis of architectures for monitoring cloud computing infrastructures," ScienceDirect, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X14002684

[9] Mohamed Mouine; Mohamed Aymen Saied, "Event-Driven Approach for Monitoring and Orchestration of Cloud and Edge-Enabled IoT Systems," IEEE Xplore Digital Library, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9860439

[10] Emmanuel Chris et al., "Cost-Benefit Analysis of Technology Implementation," ResearchGate, 2024. [Online]. Available: https://www.researchgate.net/publication/389688022_Cost-Benefit_Analysis_of_Technology_Implementation